Bigtable I/O

ng: Dataflow SDK 1.x for Java is unsupported as of October 16, 2018. After August 12, 2020, Dataflow will no
sing Dataflow 1.x and below. See Migrating from Dataflow SDK 1.x for Java
aflow/docs/guides/migrate-java-1-to-2) for migration guidance.

ocumentation on this page applies only to the Dataflow SDK 1.x for Java.

ataflow SDK 2.x for Java and the Dataflow SDK for Python are based on Apache Beam. See the documentatio
aflow/model/programming-model-beam) for those SDKs.

The Dataflow SDKs provide an API for reading data from, and writing data to, Google Cloud Bigtable (/bigtable). The `BigtableIO` source and sink let you read or write a `PCollection` of Bigtable `Row` objects from a given Bigtable.

ataflow Bigtable I/O connector is considered experimental and may change in backwards-incompatible ways i
versions of the Dataflow SDK for Java.

n also use the Dataflow HBase Connector (/bigtable/docs/dataflow-hbase), provided as part of the Bigtable
to read from and write to Bigtable in your pipeline.

Setting Bigtable Options

When you read from or write to Bigtable, you'll need to provide a table ID and a set of Bigtable options. These options contain information necessary to identify the target Bigtable cluster, including:

- Project ID
- Cluster ID
- Zone ID

The easiest way to provide these options is to construct them using `BigtableOptions.Builder` in the package `com.google.cloud.bigtable.config.BigtableOptions`:

```
tableOptions.Builder optionsBuilder =
new BigtableOptions.Builder()
    .setProjectId("project")
    .setClusterId("cluster")
    .setZoneId("zone");
```

Reading from Bigtable

To read from Bigtable, apply the `BigtableIO.read()` transform to your `Pipeline` object. You'll need to specify the table ID and `BigtableOptions` using `.withTableId` and `.withBigtableOptions`, respectively. By default, `BigtableIO.read()` scans the entire specified Bigtable and returns `PCollection` of Bigtable `Row` objects:

```
Scan the entire table.
llection <Row> btRows = p.apply("read",
 BigtableIO.read()
     .withBigtableOptions(optionsBuilder)
     .withTableId("table"));
```

If you want to scan a subset of the rows in the specified Bigtable, you can provide a Bigtable `RowFilter` object (/bigtable/docs/go/reference#Filter). If you provide a `RowFilter`, `BigtableIO.read()` will return only the `Row`s that match the filter:

```
Read only rows that match the specified filter.

Filter filter = ...;

llection <Row> filteredBtRows = p.apply("filtered read",
 BigtableIO.read()
     .withBigtableOptions(optionsBuilder)
     .withTableId("table")
     .withRowFilter(filter));
```

Writing to Bigtable

To write to Bigtable, apply the `BigtableIO.write()` transform to the `PCollection` containing your output data. You'll need to specify the table ID and `BigtableOptions` using `.withTableId` and `.withBigtableOptions`, respectively.

Formatting Bigtable Output Data

The `BigtableIO` data sink performs each write operation as a set of row mutations to the target Bigtable. As such, you must format your output data as a `PCollection<KV<ByteString, Iterable<Mutation>>>`. Each element in the `PCollection` must contain:

- The **key** of the row to be written as a `ByteString`.
- An `Iterable` of `Mutation` objects that represent a series of **idempotent row mutation operations**.

Last updated 2020-06-22 UTC.